

## **Portal Software Test Plan**

---

### **1. Scope**

---

The project, “Numerical Simulations for Active Tectonic Processes: Increasing Interoperability and Performance” has the following primary aims:

- To develop and improve the performance of applications for the simulation of earthquakes and fault systems.
- To develop and deploy federated databases for storing and retrieving both real and simulated data.
- To develop a Web Services-based interoperability structure that can be used to access applications and bind them to data sources and user interface components.
- To develop an extensible portal system that can be used to aggregate and manage user interfaces to applications, data, and miscellaneous services.

Detailed descriptions and documentation of the project may be found at <http://www-aig.jpl.nasa.gov/public/dus/quakesim/milestones.html>.

This document covers the software test plan for code developed as part of Project Milestone I, “Interoperability”. The foci of this milestone are the third and forth bullets above.

#### **1.1 System Overview**

The SERVOnGrid framework provides the interoperability that binds the QuakeSim project’s codes and data sources. SERVOnGrid consists of collection of Web services for common tasks that we bind into Application Metadata Web Services. The user interfaces for accessing these services are delivered through a browser-based Web Portal. Component interfaces are managed as “portlets” within a portal container framework. Project details may be found at <http://www-aig.jpl.nasa.gov/public/dus/quakesim/milestones.html> and <http://www.servogrid.org>.

---

### **2. Referenced Documents**

---

The following are supplemental documents that are needed to understand the test plan described herein:

- Software Development Plan: [http://www-aig.jpl.nasa.gov/public/dus/quakesim/quakesim\\_sw\\_plan20020730.pdf](http://www-aig.jpl.nasa.gov/public/dus/quakesim/quakesim_sw_plan20020730.pdf).
- Requirements Document: [http://www-aig.jpl.nasa.gov/public/dus/quakesim/CT\\_Requirements.doc](http://www-aig.jpl.nasa.gov/public/dus/quakesim/CT_Requirements.doc).
- Software Design Document: <http://www-aig.jpl.nasa.gov/public/dus/quakesim/DesignDocument2.1.doc>.
- Portal Example User Manual: <http://www-aig.jpl.nasa.gov/public/dus/quakesim/PortalExample.doc>.

---

### 3. Software Test Environment

---

**Software Version:** The system describes QuakeSim portal software v 0.31416.

**Languages/Compilers:** All software described is written in Java. All machines use the Java 2 Software Development Kit version 1.4 for compiling and running software.

**Third Party Software:** All Web software is run by Tomcat 4.0/4.1 Web servers (<http://jakarta.apache.org/tomcat/>). All Web Services use Apache Axis version 1.0 (<http://ws.apache.org/axis/>). The portal system base is built with Jakarta Jetspeed 1.4b (<http://jakarta.apache.org/jetspeed/site/index.html>). We use Apache Ant (<http://ant.apache.org/>) for both source compilation and for runtime execution management. These software packages are in turn built on several other projects; a complete list is available from the provided URLs. All third party software is freely available at no cost and open source.

**Input Data Sets:** The test portal uses GeoFEST, Disloc, and Simplex as test applications. GeoFEST input data is generated in the portal from fault and layer data in the Fault Database. Sample Simplex and Disloc input data files were provided by developers.

**Participating Organizations and Personnel:** Choonhan Youn (IU) and Marlon Pierce (IU) developed the Web service and portal software. Jay Parker (JPL) provided information and technical support for using GeoFEST and supporting codes into the portal. Peggy Li (JPL) provided information and support for using RIVA and RIVA driver scripts that were used for visualization. Anne Chen (USC) developed a Web service for accessing the fault database and provided client SQL query guidelines. Parker, Li, Theresa Baker (MIT/JPL) and Gerry Simila (Cal State Northridge) served as test users.

**Testbed Architecture and Hardware:** The portal system test environment consists of the following host machines:

1. complexity.ucs.indiana.edu: an 8 processor Sun Sunfire server. Complexity hosts the Web portal, identified as “User Interface Server” in Figure 1.
2. grids.ucs.indiana.edu: a dual processor Sun Ultra 60 server. Provides computational power for running science applications.
3. solar.uits.indiana.edu: a Sun E10000 high performance computer. See <http://www.indiana.edu/~rats/research/solar/solar.shtml> for more information. Solar provides high performance computational power for parallel applications. Solar is maintained by Indiana University’s Information Technology Services.
4. {noahsark, danube}.ucs.indiana.edu: dual processor Linux servers. These machines are used to provide computational power for running science applications.
5. jabba.jpl.nasa.gov: An 8 processor SGI maintained by Jet Propulsion Laboratory with specialized visualization software (RIVA).
6. infogroup.usc.edu: A Linux server maintained by USC that hosts the Fault database.

Unless otherwise specified, the testbed servers are owned and maintained by the Community Grids Lab.

The general architecture is depicted in the Figure 1:

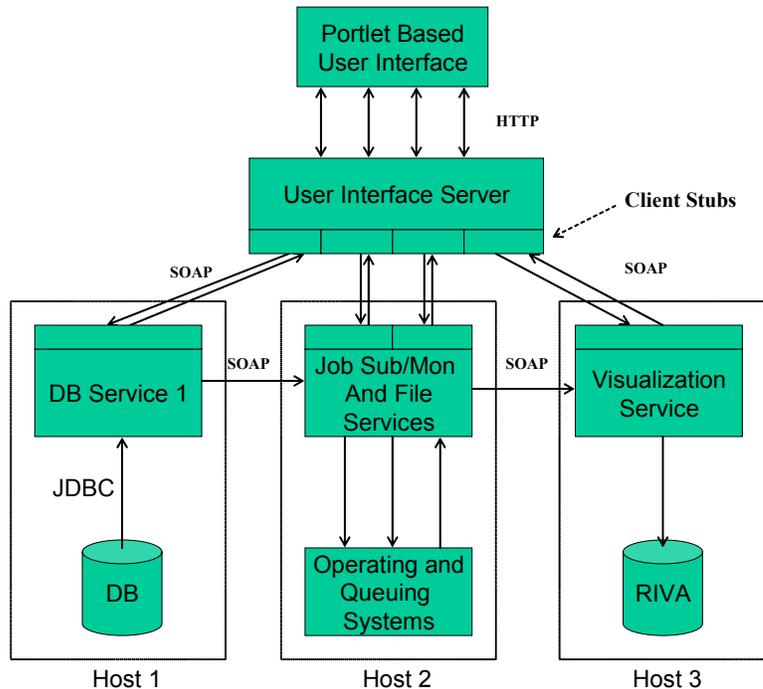


Figure 1 QuakeSim portal architecture.

The DB service is located on infogroup.usc.edu. The visualization service runs on jabba.jpl.nasa.gov. The User Interface Server is hosted on

complexity. All other listed hosts provide Job Submit/Monitor services. In the portal walkthrough described in the User Manual, danube acts as the compute host.

**Configuring the System:** The User Interface (UI) server and all backend service hosts shown in Figure 1 run Tomcat Web servers. The service hosts run Apache Axis as a web application in Tomcat. The UI server runs Apache Axis and Jetspeed as web applications. These may be configured in the standard ways, as described in the URLs above.

All system Web services that we developed may be deployed using standard methods described in the Apache Axis documentation. Clients stubs for these services may also be generated using the documentation described therein. Jetspeed portal extensions may be compiled and deployed using Apache Ant build scripts. User interface components are developed using JavaServer Pages and are deployed as a separate web application in the UI server.

---

## 4. Test Identification

---

### 4.1 General Information

Portal component testing is used to ensure each of the individual components of the system (i.e., database access or application execution) operates correctly.

Portal system tests are grouped into the following general categories:

- Positive User Testing: The portal, when used correctly in the manner prescribed in the user manual, must generate the correct final results for the selected application (GeoFEST for the tests).
- Negative User Testing: The portal should be tested for invalid user inputs and actions.
- System Testing: The portal should be tested for cases when partial system failures occur; i.e. one of the hosts or Web services is temporarily unavailable.

#### 4.1.1 Test Level

This document describes both Developer and System Integration testing.

#### 4.1.2 Test Classes

We will perform operational tests to verify component services execute as specified in design documents. We will also perform tests for invalid user inputs.

### 4.2 Planned Tests

This section describes the specific tests to be performed under this plan

#### **4.2.1 File Transfer Web Service Unit Testing**

- Purpose of the test: Verify that the file transfer web service functions (upload, download, crossload) work correctly on text and binary files, and also correctly transfers large files.
- Test Type/Class – Function/operation
- Test inputs – Test file transfer with sample Disloc input files, files in binary formats (PDF, MS Word), large files (GeoFEST output, over 68 MB).
- Verification method(s) – Files should be transferred without corruption.
- Special Requirements – None.
- Assumptions/Constraints—File transfer will not be interrupted by network problems.
- Expected results – Sample files transfer correctly.
- Actual results (added during the testing phase) – All files tested were transferred correctly.

#### **4.2.2 Job Execution Web Service Unit Testing**

- Purpose of the test: Verify that the job execution web service functions work correctly on selected QuakeSim applications (GeoFEST, Simplex, Disloc).
- Test Type/Class – Function/operation
- Test inputs – Sample input files for Disloc, Simplex, and GeoFEST, provided by application developers.
- Verification method(s) – Applications launched through the portal should produce the same output as if launched from the command line. For GeoFEST, output data should be compared to verification data.
- Special Requirements – None.
- Assumptions/Constraints—Applications must run non-interactively.
- Expected results – Output data from applications launched through the portal matches expected output data.
- Actual results (added during the testing phase) – All codes produced expected outputs.

#### **4.2.3 Context Data Management Web Service Unit Testing**

- Purpose of the test: Verify that the context data management services can be used to correctly store and retrieve arbitrary name/value pairs, and can organize these pairs in arbitrarily deep trees.
- Test Type/Class – Function/operation
- Test inputs – Application input data, fault data, and layer data all collected through HTML forms shall be stored and recovered through client interfaces.
- Verification method(s) – Demonstration for wide range of test data.
- Special Requirements – None.

- Assumptions/Constraints—Context data may be stored anywhere, but all context data for particular application is stored in the same physical location (either file system or database).
- Expected results – All data should be stored and recovered without corruption.
- Actual results (added during the testing phase) – The service behaved as expected. However, performance is poor when context data is mapped to trees that are more than one node deep (such as used in the Fault data tests). Improved design is being tested; the system interface will not change.

#### **4.2.4 Batch Script Generation Web Service Unit Testing**

- Purpose of the test: Verify that the batch script service can correctly generate C-shell and PBS scripts from user inputs. These scripts should be executed through the Job Execution service.
- Test Type/Class – Function/operation
- Test inputs –QuakeSim application input files for Disloc, Simplex, and GeoFEST are used as input to the batch service.
- Verification method(s) – Generated scripts should correctly run from the command line. PBS scripts should run correctly on PBS systems, CSH scripts should run most machines.
- Special Requirements – None.
- Assumptions/Constraints—None.
- Expected results – Scripts for selected applications should generate the expected output files.
- Actual results (added during the testing phase) –The services worked correctly when tested on grids (CSH), noahsark (PBS), and solar (PBS).

#### **4.2.5 Application Metadata Web Service Unit Testing**

- Purpose of the test: Verify that the Application Metadata Web Services (AMWS) can be used to describe a wide range of QuakeSim application code metadata and can be used to launch codes when coupled with the job execution and script generation services.
- Test Type/Class – Function/operation
- Test inputs – Application codes Disloc, GeoFEST, Simplex, and Virtual California were tested.
- Verification method(s) – Demonstration of application execution using generic services.
- Special Requirements – None.
- Assumptions/Constraints—Codes are correctly installed and running on selected hosts.

- Expected results – QuakeSim codes that were added through the application interface should be automatically added to the user interface and should be launched through the Web portal.
- Actual results (added during the testing phase) – Services worked as expected for all tested QuakeSim applications.

#### **4.2.6 Web Form Portlet Unit Testing**

- Purpose of the test: Verify that the WebFormPortlet can load remote JavaServer Pages (JSP) interfaces to the above Web services. The portlet should be loadable in Jetspeed, should maintain session states, should pass HTML Form parameters, should be navigable, and should maintain SSL connections.
- Test Type/Class – Function/operation
- Test inputs –We tested with “standalone” JSP interfaces for job submission, file transfer, and job monitoring.
- Verification method(s) –Demonstration.
- Special Requirements – None.
- Assumptions/Constraints—The JSP pages must generate valid HTML and should not depend on Javascript.
- Expected results – JSP interfaces loaded in portlets should possess the same functionality as the standalone versions. This was tested with the range of standalone interfaces described above.
- Actual results (added during the testing phase) – JSP pages worked as expected, although some incorrect HTML needed to be corrected so that the portlet could work.

#### **4.2.7 Ant Web Service Orchestration Unit Testing**

- Purpose of the test: Several QuakeSim tasks are composed of semi-independent subtasks that need to be orchestrated on one or more remote systems. We use this test to verify that Apache Ant may be launched remotely and used to execute remote tasks consisting of two or more processes, with dependencies, allowing us to treat these as single services. We made minor modifications to that Ant source code to allow it to be invoked as a Web service and wrote task extensions for generating email, performing file transfers, and interacting with messaging environments through Java Messaging Systems.
- Test Type/Class – Function/operation
- Test inputs –Sample Ant scripts that can be executed from the command line. We tested a script that can be used to generate RIVA output movies with GeoFEST input and a script that can be used to link mesh generation and boundary condition mesh decoration tools for GeoFEST.
- Verification method(s) –Demonstration.
- Special Requirements – None.

- Assumptions/Constraints—Apache Ant must be installed on the host that runs the script.
- Expected results – The Ant service should produce identical operational results as the script run from the command line (i.e. using the normal Ant interface).
- Actual results (added during the testing phase) – The services were successfully tested for both GeoFEST and RIVA. Ant services produced identical results and operation effects as scripts run from the command line.

#### **4.2.8 Database Access Web Service Unit Testing**

- Purpose of the test: Verify that the system can work with remote data services and can integrate Web services developed by other groups. This service is needed to allow other software components to access the Fault database.
- Test Type/Class – Function/operation
- Test inputs –Clients were developed to interact with the Fault database. The Fault database developer (Chen, USC) provided example SQL queries and guidelines.
- Verification method(s) –Demonstration.
- Special Requirements – None.
- Assumptions/Constraints—The Web service client is restricted to queries and is not allowed to update the database.
- Expected results – Web service clients developed by the IU team should be able to programmatically access identical information as is available through the Fault database web user interface (developed by USC).
- Actual results (added during the testing phase) –Web service clients accessed data correctly.

#### **4.2.9 Basic System Integration Testing**

- Purpose of the test: Verify the system component web services (described above) are correctly integrated through the portal.
- Test Type/Class – Function/operation
- Test inputs –Tests should be performed with GeoFEST.
- Verification method(s) –Test users.
- Special Requirements – None.
- Assumptions/Constraints—None.
- Expected results – Given instructions, users should be able to successfully create a GeoFEST input file from an initial geometry, run GeoFEST with this input file, and receive email notice when generated visualization MPEG is available for download.
- Actual results (added during the testing phase) – Tests were completed with the early testing group (Parker, Li, Baker, Simila). We

noticed during testing that application that generates the GeoFEST input file requires that the layers be added in the correct order. If this was not done, GeoFEST would run indefinitely (never completing) and so the workflow engine could not complete its execution. We also later discovered problems if the user's problem name includes spaces. In this case the portal is not able to correctly complete the mesh generation phase (the mesh generator receives an incorrect input file as input). Finally, we uncovered problems if the user provides an incorrectly specified fault (say, one of zero dimensions). The portal system again fails at the mesh generation phase (the mesh generator cannot handle a zero dimensioned fault).

---

## **5. Test Schedules**

---

The above tests were or are being conducted according to the following schedule.

- 4.2.1: Started June 1, 2002. Completed July 1, 2002.
- 4.2.2: Started June 1, 2002. Completed July 1, 2002.
- 4.2.3: Started June 1, 2002. Completed July 1, 2002.
- 4.2.4: Started June 1, 2002. Completed July 1, 2002.
- 4.2.5: Started October 1, 2002. Completed November 1, 2002.
- 4.2.6: Started November 1, 2002. Completed December 1, 2002.
- 4.2.7: Started May 12, 2003. Completed May 19, 2003.
- 4.2.8: Started May 26, 2003. Initial tests completed July 1, 2003.

---

## **6. Requirements Traceability**

---

See attached requirements traceability matrix.